On the representation of memory references generated by a program with application to the analysis of cache memories

Christine Fricker, Philippe Robert

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tel.(1) 39 63 55 11

Rapports de Recherche

N° 1158

Programme 3
Réseaux et Systèmes Répartis

ON THE REPRESENTATION OF MEMORY REFERENCES GENERATED BY A PROGRAM WITH APPLICATION TO THE ANALYSIS OF CACHE MEMORIES

Christine FRICKER
Philippe ROBERT

Février 1990

RR_1158

On the representation of memory references generated by a program with application to the analysis of cache memories

Christine Fricker and Philippe Robert

Abstract. In this paper we propose a stochastic model of the sequence of memory references generated by a program. We derive explicit expressions of the main characteristics related to such sequences: working set behavior, miss rates in the case of an architecture with a cache. The problem of (spatial/temporal) locality is addressed and the stochastic models proposed in the literature (the IRM model mainly) are discussed from this point of view. Examples with real traces generated by benchmarks of fortran programs are presented.

Sur la représentation des références mémoire générées par un programme et son application à l'étude des mémoires cache

Résumé. Nous proposons dans ce papier un modèle probabiliste de la suite des références mémoires générées par un programme. Nous donnons les formules explicites des principales caractéristiques associées à ces références: le comportement de l'ensemble de travail et le taux de défaut dans le cas d'une architecture avec cache. Le problème de la localité (spatiale-temporelle) d'un programme est étudié et les autres modèles stochastiques proposés dans ce cadre sont discutés. Nous terminons avec des exemples de traces réelles de programmes fortran.

On the representation of memory references generated by a program with application to the analysis of cache memories

Christine Fricker, Philippe Robert
INRIA, domaine de Voluceau
B.P. 105, 78153 Le Chesnay Cedex, France
fricker@seti.inria.fr, robert@solaris.inria.fr

Abstract

In this paper we propose a stochastic model of the sequence of memory references generated by a program. We derive explicit expressions of the main characteristics related to such sequences: working set behavior, miss rates in the case of an architecture with a cache. The problem of (spatial/temporal) locality is addressed and the stochastic models proposed in the literature (the IRM model mainly) are discussed from this point of view. Examples with real traces generated by benchmarks of fortran programs are presented.
Key Words: Program behavior, stochastic models, cache miss rate.
Additional Key Words: Spatial locality, working set, point processes.

1 Introduction

The design of the recent processors (such as for example the Motorola 68030 [1] and especially MIPS [10]) has shown the designers' interest in architectures with caches (on-chip caches, hierarchical caches, data caches and instruction caches, caches in multi-processors architectures...). Cache memories are small high-speed buffer memories close to the processor. They store frequently used data and instructions in order to reduce the number of accesses to the main memory. Accessing a reference in the main memory is much longer than accessing it in the cache (the ratio can vary from 4 to 10). An efficient cache is thus a crucial point in the design of a processor. To begin with, we recall some of the basic definitions concerning caches.

The cache is divided into lines, the line being the basic unit of transfer between the cache and the main memory. The line is a set of contiguous words whose size varies in general from 4 to 32 words with words of 4 bytes (depending on the total size of the cache). If a program calls a sequence of contiguous words, a miss (non presence of the reference within the cache) will possibly occur at the first access and the corresponding line will be then fetched from the main memory, all the following references will be accessed in the cache as long as the requested addresses do not overlap with the next line. This phenomena (spatial locality of successive references) is one of main motivations for using an architecture with a cache: If a line is B words long, the miss ratio (number of accesses with a miss divided by the total number of accesses) can be decreased to $1/B$ (in the cases of references to an array of integers in fortran for example). The performances of the processor will be thus boosted in the case of multiple contiguous accesses, we will focus on this point throughout the paper. As it can be guessed, the size of the line cannot be too large: the time required to transfer the line from the main memory to the cache is proportional to its length. Another reason for not increasing too much the size of the line is that the number of different parts of the program competing within the cache will increase too and thus will decrease the performances as we will see. Concerning the spatial locality of the programs, notice that some special architectures (like some supercomputers) for scientific computing do not always use a cache for the data but instead use registers to store vectors (whose components are used for a while and then discarded).

A set associative cache is divided into sets, each of them containing the same number of lines. A cache organization $\mathcal{C}$ is a triple (S, D, B) where S is the number of sets, D the number of lines per set or its degree of associativity and B the number of words per line. The cache is fully associative if $S = 1$ and direct mapped if $D = 1$. When an address ADD is requested by the processor, then (ADD right-shift BS) modulo S will correspond to the number of the set where it should be located in the cache (if the size of a set is 2^{BS} bits). The address of the beginning of each line of the set is compared with the beginning of the line corresponding to ADD. When the search through the set fails, one of the lines in the set has to be deleted. To that purpose, replacement algorithms have been proposed: Least Recently Used (LRU), FIFO, RANDOM... (see Aven [3] for a description of them).

To measure the efficiency of a cache, it is necessary to know some characteristics of the flow of references produced by the processor such as locality,

range of the memory space referenced, number of colliding lines... Our goal of this paper is to propose a mathematical model of the traces generated by a program that satisfies the following requirements, the locality described above is represented in a convenient way and *some* randomness occurs in the addresses of the references and in the way of accessing them. Due to its crucial importance in the performance evaluation of caches, representing and estimating the locality of a program is an old problem, we now review the main types of analyses in that domain.

The more direct method consists of measuring real traces of programs. The behavior of an architecture with a cache can then be analyzed through trace-driven simulations (see Smith [15] for example). One of the main characteristics obtained from real traces is the size of the working set of the program. The working set at time t $\mathcal{W}(t)$, is the set of distinct addresses accessed by the program up to time t. As a function of t, its cardinality $w(t)$ is non-decreasing, its rate of growth partially reflects the locality of the program (see Denning [7] and Coffman [5]). A recent paper Agarwal [2] proposed some other simple characteristics that can be obtained from real traces. Using them in their analytical model, the authors derive estimates of miss ratios for a given cache architecture. They introduced notions like intrinsic interference, time granule,... which we will discuss in our model. An extensive survey of the literature on the analysis of caches is included in their paper. We now turn to the mathematical models proposed in the analysis of program behavior.

The independence reference model (IRM) is the main stochastic model to represent the sequence of memory references (see Aven [3] and Rao [13]). It assumes that each reference is drawn at random according to some probability in a set $S = \{ADD_1, \ldots, ADD_N\}$ of addresses of beginning of lines, independently of the preceding references. The IRM reference model is the most frequently used in the analysis of stochastic models of replacement algorithms, mainly because of its tractability. The main difficulty with the IRM reference model is well known, the successive accesses do not have any property of locality because of the independence of the successive references (some variants like the partial Markov reference model (PMRM) assume that each reference is held for a while, but it does not really change the nature of this model). Moreover this model assumes that the size of the lines is fixed and therefore does not allow to estimate the influence of this parameter. Although it does not represent the behavior of programs with locality, the IRM model has nevertheless the interesting property that the classification of replacement algorithms according to some performance measures (miss

ratios mainly) is approximately the same for this model and real traces of various programs.

Our model of the sequence of memory references is a mixture of a local deterministic behavior, like the repeated accesses to an array in a matrix-vector multiplication and some randomness for the starting addresses, the range of the accessed references and the frequency of accesses. Our starting point has been real traces generated by two benchmarks (the Lawrence Livermore Loops and Linpacks) of fortran programs used to measure performances of computers for floating point operations. These programs present a high degree of locality, the repeated loops on vectors, matrices generate references following a rigid, periodic scheme (see Appendix) at the opposite of the IRM reference model. The model constructed from that point has then been extended to allow a more erratic behavior while keeping its initial features.

In section 2 we define and discuss the mathematical model of memory references. In section 3 the different working sets obtained with this model, the IRM reference model and real traces are compared and analyzed. Section 4 is devoted to the explicit expressions of miss rates with this model showing at the same time the tractability of this representation of memory references. In the appendix we present some examples of real traces and applications of our results.

Acknowledgments

We are indebted to Alain Lichnewsky who introduced us to the practical aspects of the subject and provided the tools to trace programs on a VAX-11/780. Thanks are due to Robert Ehrlich and William Jalby for numerous discussions on that subject.

2 Representation of the successive values of the addresses accessed by a program in the main memory

We will assume that the range of addresses in the main memory is represented by the interval $[0, M]$ for some $M > 1$ (usually $M = 2^{31} - 1$). The idea in the following is that a given program visits successively different regions of the main memory, and within each of these, there is a specific way of accessing the addresses which we will define. To illustrate this, we will use the following fortran loop:

```
DO  6  I= 2,N
J= Y(I)
X(I)= Y(I+4) + A*X(J)
6 CONTINUE
```

Every time I is incremented, the following types of references are accessed:

a) The text, the addresses concerning all the program instructions LOAD, STORE, BRANCH... and some data. The addresses referenced in this part are fixed during the whole loop.

b) The data, in this case the arrays X and Y. Two points are accessed in the Y-area `Y(I)` and `Y(I+4)` and X-area `X(I)` and `X(Y(I))`. Notice that the accesses in the Y-area are straightforward: the neighbors `Y(I+1),Y(I+5)` of `Y(I),Y(I+4)` are accessed at the next iteration, this is clearly not the case for X.

c) The stack with the state of the current process.

The total area referenced by a given program in the main memory can be split into elementary regions $(I_j)_j$ where I_j is an interval of $[0, M]$. In the above fortran loop, it will consists of some "large" intervals for the arrays X and Y and smaller one's for the different instructions and the other data (see appendix). We represent the successive memory addresses requested by a program as a sequence $(\mathcal{A}_t)_{t\geq 0}$ of subsets of the main memory $[0, M]$, where $\mathcal{A}_t$ will denote the set of addresses requested by the program during the t^{th} granule of time. A granule of time will be the duration of an elementary period during which the different regions of the main memory of the program are accessed. We will assume that the granule of time of a program is small compared to its total number of references, this will enable us to use a continuous range of values for t. In our previous example, $\mathcal{A}_t$ will be simply the values of the addresses required during the t^{th} iteration. This notion of granule differs from the notion introduced in Agarwal [2] where it is defined in terms of the start-up period of the program, in this case the value of a granule is much larger than ours.

For a fixed t, $\mathcal{A}_t = \{X^1(t), X^2(t), \ldots, X^n(t)\}$ will be a random subset of the main memory $[0, M]$, the elements $X^1(t), X^2(t), \ldots, X^n(t)$ and possibly

n are random variables with values in $\bigcup_j I_j$ and $\mathbb{N}$ respectively. For a fixed $j \leq n$, we will assume that $X^j(t)$ is in the same interval I_j for any $t \geq 0$, $X^j(t)$ being defined as an elementary access of the elementary region I_j. We now turn to the description of the function $t \rightarrow X^j(t)$, that is the way each of these elementary regions is accessed by an elementary access.

The sequence of memory references $(X(t))_{t\geq 0}$ for an elementary access within an elementary region of the main memory

Assume we want to describe the repeated accesses to some kind of arrays `Y` in a fortran loop. A natural way to proceed is to suppose that the address A_0 of `Y[0]` is some random variable with values in [0,M], that the length L of the array `Y` is also random with a probability distribution on some interval (depending on what we mean by "some kind of arrays"). From that point, no randomness occurs anymore and $Y(t)$ the value accessed at time t is defined by:

$$Y(t) = A_0 + (t \text{ modulo } L)$$

To allow a more flexible representation, we introduce some notations and definitions, but the approach is nevertheless the same as in the above simple example: a regular behavior is modulated with some randomness. As we will see, this model covers a large part of the existing representations of memory references.

Definition

An elementary access is a function $X : \mathbb{R}_+ \rightarrow [0, M]$, $t \rightarrow X(t)$, such that

. $X(T_i) = A_i \qquad i \geq 0$

. $dX(t) = \alpha dt \qquad \text{if } T_i < t \leq T_{i+1} \qquad i \geq 0$

where the A_i are some random variables on $[0, M]$ and $(T_i)_{i\geq 0}$ is a non-decreasing sequence of non-negative random variables such that $T_0 = 0$ and $lim_{i\rightarrow+\infty} T_i = +\infty$.

The T_i's are the instants of non contiguous accesses for this elementary access, at these instants the process jumps to the A_i's for $i \geq 0$. Between them the accesses are done linearly (references to contiguous words) at rate α. A region of size γ is requested for $\frac{\gamma}{\alpha}$ if no jump occurs during that time.

In the limit when $\alpha = 0$ the same address A_i is requested between T_i and T_{i+1}. With this framework, $(Y(t))_{t\geq 0}$ can be represented by

. $\alpha = 1$

. $T_i = i\,L \qquad i \geq 0$

. $A_i = A_0 \qquad i \geq 0$

An elementary region I_j will be accessed by a finite number of such accesses $X_1^j(t), \ldots, X_{q_j}^j(t)$. For example in the fortran loop at the beginning of this section there will be two accesses in the Y-area during each iteration, `Y(I)` and `Y(I+4)` . These accesses will not compete in the cache because they require the same region of the main memory. We can now define the subset $\mathcal{A}_t$ as the the union of all the values at time t of the elementary accesses of the different elementary regions:

$$\mathcal{A}_t = \bigcup_{I_j \in \mathcal{E}} \{X_k^j(t) / 1 \leq k \leq q_j\}$$

where $\mathcal{E} = \{I_j / 1 \leq j \leq n\}$ is the set of elementary regions of the program.

Examples

a) $\alpha = 1$, $A_i = X_0$ and $T_i = ir$ for $i \geq 0$.
$(X(t))_{t\geq 0}$ is simply the sequence of periodic accesses to a vector of size r. This will be the case in the above loop for the accesses to `Y(I)`.

b) $\alpha = 1$, $A_i = a_i r$ and $T_i = ir$ for $i \geq 0$ where $(a_i)_{i\geq 0}$ is some sequence of numbers with values in $\{0, 1, \ldots, r-1\}$.
For example $X(t)(t \geq 0)$ are the accesses to some rows (numbered by the a_i) of an $r \times r$ matrix. For example if $a_i = (i \text{ modulo } r)$ then the rows of the matrix are accessed in cyclic order.

c) $\alpha = 0$, the jumps between the contiguous references are defined by $T_i = i$ and $(A_i)_{i\geq 0}$ is a sequence of independent identically distributed (i.i.d.) random variables with values in a set of addresses $\{ADD_1, \ldots, ADD_N\}$. Each unit of time, a random value is accessed independently of the past. This model is exactly the IRM reference model. In the above fortran loop, the accesses to `X(Y(I))` in the X-area can be reasonably represented in that way, if we consider that the values of `Y(I)` are quite arbitrary.

d) A slight change in example c) will give the partial Markov reference model: if $T_{i+1} - T_i = 1 + g_i$ where g_i is a geometric random variable with parameter β_x conditionally on $A_i = x$. If x has been accessed at t then the next reference at $t+1$ will still be x with probability β_x. In this case, the sequence of random variables $(X(t))_{t\geq 0}$ is not an i.i.d. sequence as it is in the IRM model but a Markov chain.

e) $\alpha = 0$, $T_i = i$ and $A_i = \sum_{k=0}^{i} Y_i$ where $(Y_i)_{i\geq 0}$ is a sequence of i.i.d. random variables with the common distribution $P(Y_0 > x) = \left(\frac{x}{x_0}\right)^{-\theta}$ on $[x_0, +\infty[$. The accesses $X(t)$ $(t \geq 0)$ in this case are the one's described in Thiebaut [16] where a model using fractal geometry is proposed.

Remarks

1) The examples a) and b) are "deterministic", they are typical of fortran loops on arrays (cf Appendix).

2) We have chosen a continuous model for the range of values of the addresses for simplicity of presentation, the usual definitions concerning caches will be modified accordingly. If we compare the size of a word with the total size of the main memory, it is a reasonable assumption.

Our analysis of this model will not cover all the generality of this model (especially the cases c) and d) which are already known). Nevertheless our approach can be extended to most of the cases.

3 The size of the working set of a program

In this section, we will assume that the parameters of an elementary access $X(t)$ are such that $(A_i)_{i\geq 0}$ and $((T_{i+1} - T_i)_{i\geq 0}$ are independent sequences of i.i.d. random variables and $\alpha = 1$. The working set of an elementary access at time t will be

$$\mathcal{W}(t) = \bigcup_{k\leq i} [A_{k-1},\, A_{k-1} + T_k - T_{k-1}] \bigcup [A_i,\, A_i + t - T_i]$$

for $T_i \leq t < T_{i+1}$.

The size of the working set at time t, $w(t)$ will be the Lebesgue measure of $\mathcal{W}(t)$:

$$w(t) = \int_0^M 1_{\{x\in\{X(s)/s\leq t\}\}}\, dx$$

where M is the total size of the main memory.

Proposition 3.1 *The Laplace transform* $\widetilde{W}(\lambda) = E\left(\int_0^{+\infty} w(t)e^{-\lambda t}\,dt\right)$ *of the mean size of the working set size can be expressed as:*

$$\widetilde{W}(\lambda) = \frac{M}{\lambda} - \int_0^M \frac{E\left(1 - e^{-\lambda(x-A_0)\wedge T_1} 1_{\{A_0 \le x\}} - e^{-\lambda T_1} 1_{\{A_0 \ge x\}}\right)}{\lambda(1-\phi(\lambda,x))}\,dx$$

with $\phi(\lambda,x) = E(e^{-\lambda T_1}(1 - 1_{\{0 \le x - A_0 \le T_1\}}))$, $E(Z)$ *denotes the expected value of* Z *and* $a \wedge b = Min(a,b)$.

Proof

According to the definition of an elementary access,

$$X(t) = \sum_{i=0}^{+\infty} (A_i + t - T_i)\, 1_{\{t \in [T_i, T_{i+1}]\}}$$

define

$$\overline{W}_x(t) = P(x \notin \bigcup_{T_i < t} [A_i, A_i + t \wedge T_{i+1} - T_i])$$

then

$$\overline{W}_x(t) = E\left(\sum_{i=0}^{+\infty} 1_{\{T_i \le t < T_{i+1}\}} \prod_{k=0}^{i-1} 1_{\{x \notin [A_k, A_k + T_{k+1} - T_k]\}} \times 1_{\{x \notin [A_i, A_i + t - T_i]\}}\right)$$

after some routine calculations using that

$$\widetilde{W}(\lambda) = E\left(\int_0^{+\infty} w(t)e^{-\lambda t}\,dt\right) = \frac{M}{\lambda} - \int_0^M dx \int_0^{+\infty} \overline{W}_x(t)e^{-\lambda t}\,dt$$

we obtain the desired formula.

Comparison with the rate of growth of the IRM model

Consider the following elementary access

$$X(t) = t \text{ modulo } L$$

where L is some random variable with distribution function F on $[0,1]$. If we look at the statistics of this process then the interval $[x, x+dx]$ is visited with probability $h(x)dx$ with $h(x) = \frac{1}{L} 1_{[0,L]}(x)$. Now denote by $(Y_n)_{n \ge 0}$ the sequence of accesses of an IRM reference model such that Y_n is the

reference requested at time $t = n\gamma$, Y_n is chosen at random with respect to the probability $(p_i)_i$ such that

$$p_i = P(Y_n = i\gamma) = \int_{i\gamma}^{(i+1)\gamma} h(u)\,du$$

for $i = 0, \ldots, \frac{1}{\gamma} - 1$. Clearly when γ is small, this IRM model will have the same statistics as $X(t)$. Denote by $W_Y(t)$ and $W_X(t)$ the corresponding mean sizes of the working sets at time t. With the above definition of the size of the working set

$$W_X(t) = \int_0^1 t \wedge x\, F(dx)$$

if $w_Y(t)$ is the mean size of the working set for a fixed L, using that

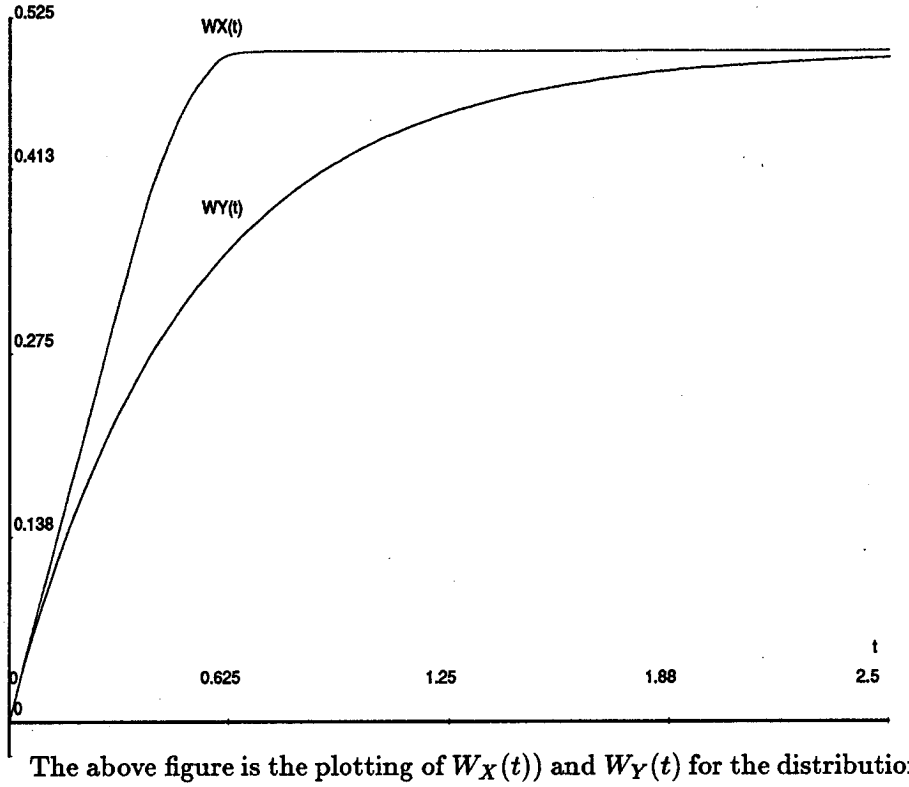
$$\begin{aligned} w_Y(t) &= \gamma \sum_{i=0}^{\gamma^{-1}-1} (1 - P(Y_k \neq i\gamma,\ \forall k\, k\gamma \leq t)) \\ &= \gamma \sum_{0}^{\gamma^{-1}-1} (1 - (1-p_i)^{\frac{t}{\gamma}}) \end{aligned}$$

some straightforward calculations show that when $\gamma \to 0$ then

$$w_Y(t) = 1 - \int_0^1 e^{-th(x)} dx$$

integrating with respect to L and using that $1 - e^{-x} \leq x \wedge 1$, we get

$$W_Y(t) = \int_0^1 x(1 - e^{-\frac{t}{x}})\, F(dx) \leq W_X(t).$$



The above figure is the plotting of $W_X(t))$ and $W_Y(t)$ for the distribution function $F(dx) = 3 \times 1_{[1/3,2/3]}(x)dx$. Notice that the growth of $W_X(t)$ is greater for small values of t. We choose this simple example because it is representative a general phenomenon encountered in the traces of the programs we analyzed, in these cases the IRM reference model underestimates the real size of the working set (see Appendix). The reason is quite simple: when an array is referenced, until the end of the array a new reference is accessed, a linear growth is thus expected during that time. In the IRM model, the probability of having an "old" reference increase with time and then a sublinear growth is expected for the for the size of the working set.

4 Miss ratios, analytical expressions

We assume that the cache is represented by the torus $[0, C]$ for some $C > 0$. Thus an address ADD of the main memory has to be reduced modulo C in order to get its location in the cache. We will assume throughout this section that the beginnings of elementary regions are uniformly distributed

over $[0, C]$ and independent. According to Agarwal [2], this is a reasonable assumption when C is not too large. Nevertheless the asymptotic formulas obtained in this section can be extended to the case where the distribution of the beginning of elementary regions is not uniform (see Fricker [9]). For simplicity of presentation we will assume that each elementary access $X^j(t)$ has the following characteristics:

. $A_i^j = U_j \qquad i \geq 0$

. $T_i^j = i\,L_j \qquad i \geq 0$

. $\alpha = 1$

where U_j is a uniform r.v. on $[0, C]$ and L_j a random variable with some distribution $F(dx)$ on $[0, C]$. The elementary access $X^j(t)$ is a repeated reference to the interval $I_j = [U_j,\, U_j + L_j]$, $X^j(t) = U_j + (t \text{ modulo } L_j)$.

The variable L_j is the length of contiguous accesses of the j^{th} elementary access, its distribution function F will be our main parameter concerning the locality of the program: typically it has a peak for small values (for the instructions and some data) and, if it is the only peak, only small non contiguous parts of the main memory are accessed thus entailing a temporal locality (in our model an interval of size x is referenced t/x times between 0 and t). Otherwise if the program accesses some large arrays, there will be other peaks around some other values for F and thus some spatial locality (contiguous words are accessed between the beginning and the end of the interval). Notice that we assumed that $L_j \leq C$ for any j so that the cache cannot be completely flushed by any elementary access.

According to the terminology of Agarwal [2], the set $\bigcup_{i \neq j} I_i \cap I_j$ is the potential collision set. The main difference in our model is the tight coupling between the frequencies of collisions and the collision set itself, the sizes of the intervals and their places in the cache will determine them as we will see. In Agarwal [2] there is no relation between the frequencies and the potential collision set.

In the next two sections we derive analytical expressions for the miss ratios for that model of memory references. In the first section we focus on the influence of our parameter of locality the distribution function F. For that purpose we assume that the size of the line is small compared to the size of the cache, this enables us to use a continuous model. The expression

obtained for the miss ratio is a simple functional of F. In the second section we deal with the general case, i.e. a line of size l, the expressions for the miss ratios are less simple as functionals of l and F but usable by any symbolic computation package. To our knowledge the analytical expressions of miss ratios involving the line size as a parameter are very rare. The size of the line is usually strongly related to the model of memory references used (as in the IRM model for example) and thus cannot be used as a parameter.

4.1 The case of a direct mapped cache with a small size of line

In this section, we consider the case where the size of the line is small compared to the size of the cache. This assumption allows the use a continuous model in the following sense, if $X^j(t) = x$ for some j and if j is not the last access that referenced x, then during t and $t + dx$ the j^{th} access will have to fetch the part of I_j in the main memory corresponding to $[x, x + dx]$ in the cache. This leads us to the following definition of the miss ratio for this section,

Definition

The miss ratio due to an elementary access at time t is the proportion of time between 0 and t that it fetches its references directly in the main memory. The miss ratio of a program will be the average of the miss ratios of its elementary accesses.

We suppose that we have n independent elementary accesses $(X^j(t))_{1 \leq j \leq n}$ as described above with the associated independent sequences of i.i.d. r.v. $(U_j)_{1 \leq j \leq n}$, $(L_j)_{1 \leq j \leq n}$. The j^{th} elementary access is simply

$$X^j(t) = U_j + (t \text{ modulo } L_j) \qquad \textit{for } t \geq 0.$$

The misses are of two types:

a) The misses due to the start-up phase when the intervals $[U_j, U_j + L_j]$ are referenced for the first time. Between $t = 0$ and $t = L_j$, the j^{th} elementary access will fetch its references in the main memory. Thus the miss rate at time t due to the start-up phase will be simply:

$$D_{st}(t) = \frac{1}{nt} \sum_{j=1}^{n} L_j \wedge t$$

with $a \wedge b = \text{Min}(a, b)$.

b) The misses due to conflicts with the other accesses, because of some overlapping of the different intervals within the cache. If $X^j(t) = x$ we know that $X^j(t - L_j) = x$ and that there will be a conflict if some other access hits x between $t - L_j$ and t. In this case, because of the direct mapped architecture, the j^{th} access will have to fetch in the main memory the reference which is mapped in x in the cache.

Let $d_x^j(t)$ denote the number of such conflicts between 0 and t. Then the miss rate due to the conflicts is

$$D_{cf}(t) = \frac{1}{n}\sum_{j=1}^{n} \frac{1}{t}\int_0^C d_x^j(t)\,dx.$$

The miss ratio $D(t)$ will be then defined as the sum of these two miss ratios.

Proposition 4.1 *The mean miss ratios at time t of n elementary accesses on a cache of size C are given by*

$$E(D_{st}(t)) = \frac{1}{t}\int_0^{t\wedge C} x\,F(dx) + F([t \wedge C,\, C])$$

for the start-up period and

$$E(D_{cf}(t)) = \frac{1}{t}\int_0^C \int_0^x \left[\frac{(t-u)^+}{x}\right] du \left(1 - \left(1 - \frac{1}{C}\int_0^C x \wedge y\,F(dy)\right)^{n-1}\right) F(dx)$$

for the collision of addresses within the cache. In particular the stationary miss ratio is given by

$$1 - \int_0^C \left(1 - \frac{1}{C}\int_0^C x \wedge y\,F(dy)\right)^{n-1} F(dx)$$

where $F(dx)$ is the distribution function of the size of elementary regions, $[a]$ is the integral part of a, $a^+ = Max(0, a)$ and $a \wedge b = Min(a, b)$.

Proof

The expression for $E(D_{st}(t))$ is clear from the above definition. We will assume for the moment that the values of $L_1 \ldots L_n$ are fixed and that the expected values concern only the U_j's. If $x \in [0, C]$ then $N_x^j = (t_{1,x}^j + kL_j)_{k\geq 0}$ is the sequence of the hitting times of x by $(X^j(t))_{t\geq 0}$ if $t_{1,x}^j = \inf\{t/X^j(t) = x\}$, with the convention $\inf\{\emptyset\} = +\infty$ and $N_x^j \equiv 0$ in this case. Notice that

$$d_x^j(t) = \sum_{s\in N_x^j, s+L_j<t} 1_{\{N_x^k[s,s+L_j]\neq 0 \,for\ some\, k\neq j\}}$$

where $N_x^j[a,b]$ is the number of points of N_x^j between a and b.

As the law of U_j is uniform on $[0,C]$, the law of $t_{1,x}^j$ conditioned on the event $N_x^j \not\equiv 0$ is a uniform distribution on $[0, L_j]$. Thus conditioned on $N_x^j \not\equiv 0$ the point process N_x^j is stationary: its law is invariant under any x-shift for $x > 0$ (see Neveu [12]). As the empty point process is of course stationary, we deduce that N_x^j is a stationary point process for any $j \leq n$.

Using the homogeneity property which is a consequence of the uniform distribution of the random variables $(U_j)_{1\leq j\leq n}$ we can take $x = 0$ and forget the subscript x. The expected value of $D_{cf}(t)$ is thus

$$C\frac{1}{n}\sum_{j=1}^{n}\frac{1}{t}E\left(\sum_{s\in N^j, s+L_j<t} 1_{\{N^k[s,s+L_j]\neq 0 \text{ for some } k\neq j\}}\right).$$

The number of points of N^1 with a right neighbor in $[0,\, t]$ is $\left[\frac{(t-t_1^1)^+}{L_1}\right]$. By symmetry we will compute the first term in the previous sum; using the independence of the U^j's and the stationarity of the N^j's, this term can be expressed as

$$\frac{1}{t}E\left(\left[\frac{(t-t_1^1)^+}{L_1}\right]\left(1-\prod_{j=2}^{n}P(N^j[0,L_1]=0)\right)\right.$$

but

$$\begin{aligned} P(N^j[0,\, L_1]=0) &= P(N^j\equiv 0)+P(N^j\not\equiv 0)P(t_1^j > L_1/N^j\not\equiv 0) \\ &= 1-\frac{L_j}{C}+\frac{(L_j-L_1)^+}{C} \\ &= 1-\frac{L_j\wedge L_1}{C}. \end{aligned}$$

Integrating with respect to the L_j's for $j \geq 2$ we get,

$$\frac{1}{tC}\int_0^{L_1}\left[\frac{(t-u)^+}{L_1}\right]du\left(1-\left(E(1-\frac{L_2\wedge L_1}{C}/L_1)\right)^{n-1}\right)$$

and finally we obtain the expression for $D_{cf}(t)$. The stationary miss ratio is obtained by letting $t \to +\infty$. □

Remark

The expression obtained for the stationary miss rate is a simple functional of our parameter of locality of the program, the distribution function $F(dx)$.

It is remarkable that in this case the transient behavior of the cache can be expressed explicitly, and thus the effects of a multiprogramming environment on the cache (see Fricker [9]). Although the expressions for the stationary miss ratios are known in the IRM reference model, they are quite complicated functionals of the probabilities $p_1, \ldots, p_N$ and become hard to use in practice when the number of distinct references N is large (cf Aven [3] and Flajolet [8]).

This proposition can be generalized in the following way, we still have n elementary distinct intervals $I_j = [U_j, U_j + L_j]$ with the same hypothesis on the random variables $(U_j)_j$ and $(L_j)_j$. We assume that the interval I_j is accessed by q_j elementary accesses $X_i^j(t)$ $i = 1, \ldots, q_j$ which do not compete within the cache because they reference the same region of the main memory. Such a situation has been encountered in our fortran example, two accesses `Y(I)` and `Y(I+4)` are not competing within the cache. The beginnings of the q_j accesses $(X_i^j(0))_{1 \leq i \leq q_j}$ are assumed to be independent and uniformly distributed on I_j. We have the following proposition,

Proposition 4.2 *The stationary miss ratio of a program of n elementary regions and q_j elementary accesses for the j^{th} region is given by*

$$\frac{\sum_{j=1}^{n} q_j \left(1 - \int_0^C \prod_{i \neq j} \left(\int_0^C \left(1 + x \frac{((\frac{y-x}{x})^+)^{q_j} - 1}{q_j C}\right)^{q_i} F(dy)\right) F(dx)\right)}{\sum_{j=1}^{n} q_j}.$$

sketch of the proof

As before fix for the moment the values of $L_1 \ldots L_n$. According to the previous proof, it is sufficient to look at the misses caused by the q_1 elementary accesses to 0 of the elementary region 1. Let $X^1(t)_{t \geq 0}$ one of these elementary accesses. If $X^1(t) = 0$ then $X^1(t - L_1) = 0$ but there are also $q_1 - 1$ accesses of region 1 to 0 between $t - L_1$ and t and because of the independence assumptions they occur uniformly and independently between $t - L_1$ and t. Thus an elementary access from region $j \neq 1$ will cause a miss only if it hits 0 between the maximum of these $q_1 - 1$ variables and t. This probability of not having a miss can be obtained as

$$1 + L_1 \frac{\left(\left(\frac{(L_1 - L_j)^+}{L_1}\right)^{q_1} - 1\right)}{C q_1}$$

remembering that we have q_j such independent such accesses for the region j, the formula is then quite easy to obtain. □

The transient behavior can also be obtained in this case.

4.2 The influence of the size of the line in a direct mapped cache

In this section we denote by w, l, C the respective sizes of a word, a line and the cache. We still assume that we have a program with n elementary accesses. If an elementary interval is mapped onto $[kw, Kw]$ in the cache, such that $kw \leq Kw$, the access requires successively the $K-k$ words $[kw,(k+1)w],\ldots,[(K-1)w,Kw]$. If there is a miss when this access reaches the words which are mapped in some line $[ml,(m+1)l] \subset [kw,Kw]$ of the cache, then the whole line has to be fetched from the main memory. This line will have to be in the cache at each of $\frac{l}{w}$ accesses of the words $[ml, ml+w],\ldots,[(m+1)l-w,(m+1)l]$.

As in the proofs of the previous propositions, we will denote $N^j = (t_i^j)_{i\geq 0}$ the instants where the j^{th} access require some fixed line $\mathcal{L}$ (say $[0,l]$) in the cache, with the same convention as before $N^j \equiv 0$ and $t_1^j = +\infty$ if $\mathcal{L}$ is not accessed by j. The periodic structure already encountered is still valid in the sense that if $t \in N^j$ then $t + L_j \in N^j$. The main difference here is that the accesses to $\mathcal{L}$ are done by "bursts", if $t \in N^j$ is an instant where the first word of $\mathcal{L}$ is requested then $t+w, t+2w,\ldots,t+mw$ will be also in N^j, for some m depending on the number of words in $\mathcal{L} \cap I_j$. After these instants there is a "gap" in N^j until $t + L_j$. Notice that in the previous section, the hypothesis of a small size of line permitted to reduce the bursts $t+w, t+2w,\ldots,t+mw$ to one point.

The direct mapped architecture for a cache has the advantage of simplicity for the search of an address. There is only one line to check in the cache to know if the line has not been erased by another access. It has nevertheless the following drawback that, if two different accesses require the same line of the cache at the same time, then each word accessed within the line will cause a miss. For example, take $w = 1$, $l = 4$, two accesses a and b and the following sequence of the accesses

$$(0,a,4),(1,b,3),(2,a,5),(3,b,4),(4,a,6),(5,b,5),(6,a,7),(7,b,6),(8,a,8)\ldots$$

where the first coordinate is the time of the access and the last one, the word required. At time $t = 3,5,7$, the line $[4,8[$ is accessed by b and at time $t = 4,6$ by a and at each of these instants the owner of the line alternates between a and b entailing a miss at each word accessed. Notice that if the cache is associative and that the set size is two lines so that two different lines can be present in the same set at the same time, the same phenomena

can occur but it will require that three accesses are in the same set at the same time which is more unlikely.

In order to study this additional phenomena, we will decompose the misses into two categories. The misses of type 1, the "normal" misses eventually occur at the entrance in a line (at time $t = 0, 3$ in our example). The misses of type 2 will be the other misses, when the previous word accessed is in the line and there is a miss for the next word in this line (at time $t = 4, 5, 6, 7$).

We will assume that the starting point of I_j is uniformly distributed on $[0, C]$ and that the distribution of L_j has a density $h(x)$ on $[0, C]$. The j^{th} access requires a line $\mathcal{L}$ at $t_1^j, t_1^j + w, \ldots, t_1^j + q_j, t_1^j + L_j, t_1^j + L_j + w, \ldots$ and q_j is the random variable $\left[\frac{|I_j \cap \mathcal{L}|}{w}\right] w \wedge l$ conditioned on $I_j \cap \mathcal{L} \neq \emptyset$ (where $|S|$ is the length of the set S) and t_1^j the very first access of $\mathcal{L}$ is infinite if $I_j \cap \mathcal{L} = \emptyset$. The misses of type 1 can therefore occur only at the $t_1^j + kL_j$, $k \geq 0$. To reduce the complexity of the expressions in the next proposition for the miss ratios, we will assume that $h(x) = 0$ for $x \in [C - l, C]$, i.e. $L_j + l \leq C$ almost surely for any j.

Proposition 4.3 *The stationary miss rate D of a program of n elementary accesses in a direct mapped cache of size C with lines (resp. words) of size l (resp. w) can be decomposed in $D_1 + D_2$ with*

$$D_1 = \int_{[0,C]\times\mathbb{R}} \frac{w}{\eta + w} \left(1 - \left(1 - \int_{[0,C]\times\mathbb{R}} \frac{l+y}{C}(1 \wedge \frac{x - \eta + \nu}{y})\, \Phi(dy, d\nu)\right)^{n-1}\right) \Phi(dx, d\eta)$$

where D_1 is the miss rate due to the accesses of the first words in a line of the cache for an elementary access, and

$$D_2 = \left(\int_{[0,C]\times\mathbb{R}} \frac{\eta}{\eta + w}\, \Phi(dx, d\eta)\right) \left(1 - \left(1 - \int_{[0,C]\times\mathbb{R}} \frac{l+y}{C}(1 \wedge \frac{w + \nu}{y})\, \Phi(dy, d\nu)\right)^{n-1}\right)$$

where D_2 is the miss rate due to the accesses which follow an access in the same line, $h(x)$ is the density of the size of an elementary region and $\Phi(dx, d\eta)$ is the probability distribution of (L_1, q_1) on $[0, C] \times \mathbb{R}$ defined by

$$\int_{[0,C]\times\mathbb{R}} F(x, \eta)\, \Phi(dx, d\eta) = \int_l^C \left(\frac{x - l}{x + l} F(x, l) + \sum_{k=0}^{[\frac{l}{w}]-1} \frac{2w}{x + l} F(x, kw)\right) h(x)\, dx$$

$$+\sum_{k=0}^{[\frac{l}{w}]-1}\int_{kw}^{(k+1)w}\left(\frac{l+x-2kw}{l+x}F(x,kw)+\sum_{i=0}^{k-1}\frac{2w}{l+x}F(x,iw)\right)h(x)\,dx.$$

Proof

As in the previous proofs we fix the values of $L_1,\ldots,L_n$ but also the starting points of the I_j's. It is sufficient to look at the misses due to the access of $\mathcal{L}=[0,l[$ by the first access. According to the previous definitions, the number of misses of type 1 for the access of $\mathcal{L}$ at time t is

$$d_1(t)=\sum_{k;t_1^1+kL_1\leq t}1_{\{N^i[t_1^1+(k-1)L_1+q_1,t_1^1+kL_1]\neq 0,\,for\ some\, i\neq 1\}}$$

because $t_1^1+(k-1)L_1+q_1$ is the last access in $\mathcal{L}$ by 1 before $t_1^1+kL_1$. If $(t_1^1-t_1^i+kL_1)\bmod L_i\geq L_1-q_1+q_i$ then for some integer a

$$t_1^i+aL_i+q_i\leq t_1^1+(k-1)L_1+q_1\leq t_1^1+kL_1\leq t_1^i+(a+1)L_i$$

in other words, there are no access of $\mathcal{L}$ by i between $t_1^1+(k-1)L_1+q_1$ and $t_1^1+kL_1$ and thus no miss at $t_1^1+kL_1$ due to i. We can rewrite $d_1(t)$ as

$$d_1(t)=\sum_{k\geq 0,t_1^1+kL_1<t}\left(1-\prod_{i=2}^{n}1_{\{(t_1^1-t_1^i+kL_1)\bmod L_i\geq L_1-q_1+q_i\}}\right)$$

$$d_1(t)=\sum_{k=0}^{l_1(t)}\left(1-\prod_{i=2}^{n}1_{\{\frac{t_1^1-t_1^i+kL_1}{L_i}\bmod 1\geq\frac{L_1-q_1+q_i}{L_i}\}}\right)$$

where $l_j(t)=card\{k\geq 1/t_1^j+kL_j<t\}$ for $j=1\ldots n$, the miss rate for the first access will be $\frac{d_1(t)}{l_1(t)}$. Denote by $\alpha=((\alpha_i)_{1\leq i\leq n-1})$ with $\alpha_i=\frac{L_1}{L_{i+1}}$ and T the translation on the torus $[0,1]^{n-1}$, $T((x_i)_i)=((x_i+\alpha_i\bmod 1)_i)$, then $d_1(t)$ can be rewritten as

$$d_1(t)=\sum_{k=0}^{l_1(t)}f(T^k(y)),$$

if $y=(\frac{t_1^1-t_1^2}{L_2},\ldots,\frac{t_1^1-t_1^n}{L_n})$, $T^k=\underbrace{T\circ\ldots\circ T}_{k\ times}$ and $f(x)=1-\prod_{i=1}^{n-1}1_{\{x_i\geq\frac{L_1-q_1+q_{i+1}}{L_{i+1}}\}}$.

A classical theorem of ergodic theory on the invariant measures of translations of the torus (see Cornfeld [6] for example) ensures that

$$\lim_{l_1\to+\infty}\frac{1}{l_1}\sum_{k=1}^{l_1}f(T^k(x))=\int_{[0,1]^{n-1}}f(u)\,du_1\ldots du_{n-1},$$

Lebesgue almost surely on $[0,1]^{n-1}$, provided that the $\alpha_1,\ldots,\alpha_{n-1}$ are rationally independent, i.e. if for some integers $k_0,\ldots,k_{n-1}$ the relation $k_0+\sum_{i=1}^{n-1} k_i\alpha_i = 0$ holds, then $k_i = 0$ for $i = 0,\ldots,n-1$ (the vectors $1,\alpha_1,\ldots,\alpha_{n-1}$ are free in the Z-module $\mathbb{R}$). This property is true in our case, L_j, $j = 1..n$ are almost surely rationally independent because their law is absolutely continuous with respect to the Lebesgue measure. Gathering these results we obtain that for fixed L_j, q_j $j = 1,\ldots,n$

$$\lim_{t\to+\infty}\frac{d_1(t)}{l_1(t)} = 1-\prod_{i\neq 1}(1_{\{N^i\equiv 0\}}+1_{\{N^i\not\equiv 0\}}(1-\frac{L_1-q_1+q_i}{L_i})^+).$$

Integrating with respect to the events $I_j\cap\mathcal{L}=\emptyset$, $j=2,\ldots,n$, the right hand side becomes

$$1-\prod_{i\neq 1}(1-\frac{l+L_i}{C}+\frac{l+L_i}{C}(1-\frac{L_1-q_1+q_i}{L_i})^+).$$

Using that the ratio of the number of accesses to the first word and the total number of accesses to this line converges to $\frac{w}{q_1+w}$, integrating again, we get that the stationary miss rate of type 1 for $\mathcal{L}$ is

$$E\left(\frac{w}{w+q_1}\left(1-\left(E(1-\frac{l+L_2}{C}(1\wedge\frac{L_1-q_1+q_2}{L_2})/L_1,q_1)\right)^{n-1}\right)\right),$$

and thus the first part of our proposition. For the misses of type 2 the proof follows exactly the same lines, starting from the following expression for the number of misses of type 2 up to time t,

$$d_2(t) = \sum_{s=t_1^1+kL_1+k'w,k\geq 0,0<k'\leq q_1,s\leq t} 1_{\{N^i[s-w,s]\neq 0,\,for\ some\, i\neq 1\}}.$$

□

5 Conclusion

The initial motivation of this paper was to propose a mathematical description of the traces generated by a program so that the influence of the different parameters of a cache could be estimated. We have shown in the previous sections that our model had some of the properties usually encountered in programs and was mathematically tractable. For sake of simplicity we dealt

only with direct-mapped caches, thus the parameter of associativity did not appear, nevertheless the same kind of analysis can be done in the case of an associative cache provided it is maintained with the LRU replacement policy (see Fricker [9]). The other point where our initial program is not completely filled is the regularity of the programs we represented, we think that the representation we gave is quite representative of programs used in scientific computing where this regular behavior was observed (the absence of this behavior is in our opinion, the main flaw of the previous stochastic models). Nevertheless according to the experiments we made on some Unix functions (like ld, compress, a C compiler...) which are less regular, some randomness has to be introduced in the way of accessing the elementary regions and in the number of elementary accesses (see Fricker [9]). Finally, another important point has been neglected in our paper and in almost all the analytical models of caches, it is the influence of the values of the different access times (to the cache, to the main memory and possibly to an intermediate cache between them). These parameters strongly influence the design of a cache (a glance at the proceedings of the conference on computer architecture of 1989 is sufficient to be convinced how crucial it is), but up to now they have not been included in the analytical models of caches.

References

[1] *First look at motorola's latest 32-bit processor*, Electronics, (1986), pp. 71–75.

[2] A. AGARWAL, M. HOROWITZ, AND J. HENNESSY, *An analytical cache model*, ACM Trans. Comput. Syst., 7 (1989), pp. 184–215.

[3] O. I. AVEN, E. G. COFFMAN, AND Y. A. KOGAN, *Stochastic Analysis of computer storage*, Reidel, Amsterdam, 1987.

[4] M. BADEL AND J. LEROUDIER, *Performance evaluation of a cache memory for a mini-computer*, Tech. Rep. 335, IRIA, Domaine de Voluceau, Rocquencourt, BP 105 78153 Le Chesnay Cedex (France), Dec. 1978.

[5] E. G. COFFMAN AND P. J. DENNING, *Operating Systems Theory*, Prentice-Hall, 1973.

[6] I. P. CORNFELD, S. V. FOMIN, AND Y. G. SINAI, *Ergodic Theory*, vol. 245 of Grundlehren der mathematischen Wissenschaften, Springer Verlag, 1982.

[7] P. J. DENNING, *The working set model for program behavior*, Comm. ACM, 11 (1968), pp. 323–333.

[8] P. FLAJOLET, D. GARDY, AND L. THIMONIER, *Birthday paradox, coupon collectors, caching algorithms and self-organizing search*, Tech. Rep. 720, INRIA, Domaine de Voluceau, Rocquencourt, BP 105 78153 Le Chesnay Cedex (France), Aug. 1987.

[9] C. FRICKER AND P. ROBERT, *A stochastic model of memory references.* in preparation, Oct. 1989.

[10] M. HOROWITZ, P. CHOW, D. STARK, R. T. SIMONI, A. SALZ, S. PRZYBYLSKI, J. HENNESSY, G. GULAK, A. AGARWAL, AND J. ACKEN, *MIPS-X: A 20-MIPS peak, 32-bit microprocessor with on-chip cache*, IEEE of Solid-State Circuits, 22 (1987), pp. 790–799.

[11] K. MEVISSEN, *Untersuchungen zur lokalitat in technisch-wissenschaftlichen programmen*, tech. rep., KFA, Zentralinstitut fur Angewandte Mathematik, Mar. 1987.

[12] J. NEVEU, *Processus ponctuels*, vol. 598 of Lecture notes in maths, Springer Verlag, Berlin, 1976.

[13] G. S. RAO, *Performance analysis of cache memories*, J. Assoc. Comput. Mach., 25 (1978), pp. 378–395.

[14] A. SCHROEDER, *A statistical approach to the study of program behavior via reference string analysis*, in Computer Performance, K. M. Chandy and M. Reiser, eds., North-Holland, 1977, pp. 381–196.

[15] A. J. SMITH, *Cache memories*, ACM Comput. Surveys, 14 (1982), pp. 473–530.

[16] D. THIEBAUT, *On the fractal dimension of computer programs and its application to the prediction of the cache miss ratios*, IEEE Trans. Comput., 38 (1989), pp. 1012–1026.

Appendix

Examples of traces, the addresses requested are represented with the time at which they are requested on the horizontal axis.

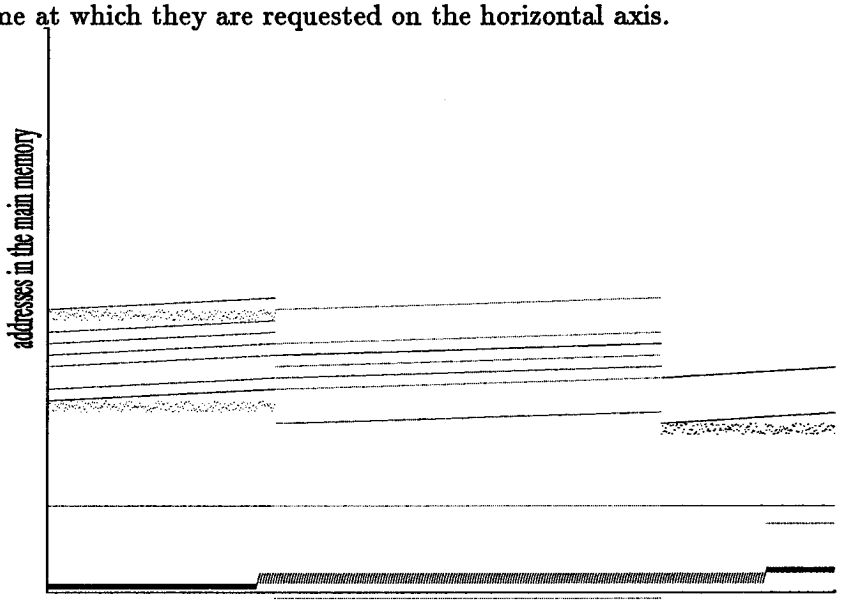


figure-1 9000 references of the Lawrence Livermore loops, kernel 14

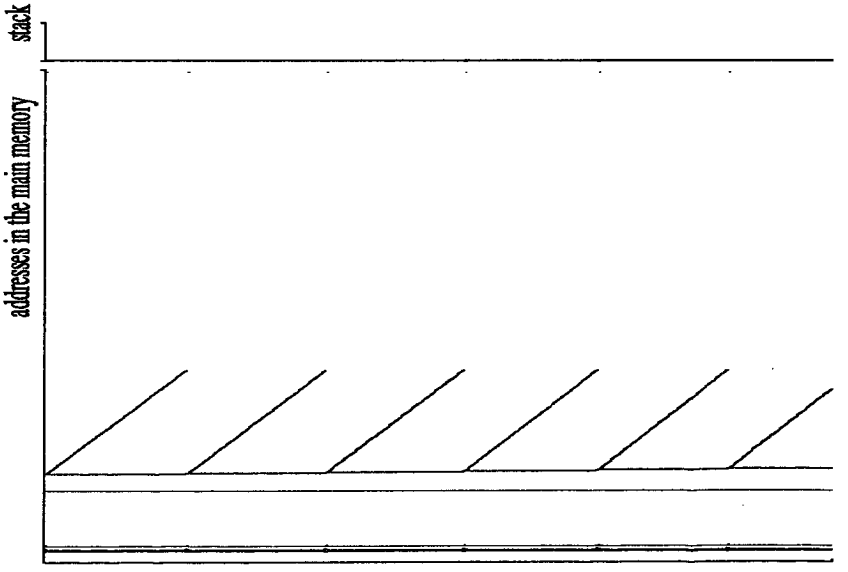


figure-2 10^6 references of the fortran benchmark LINPACKS

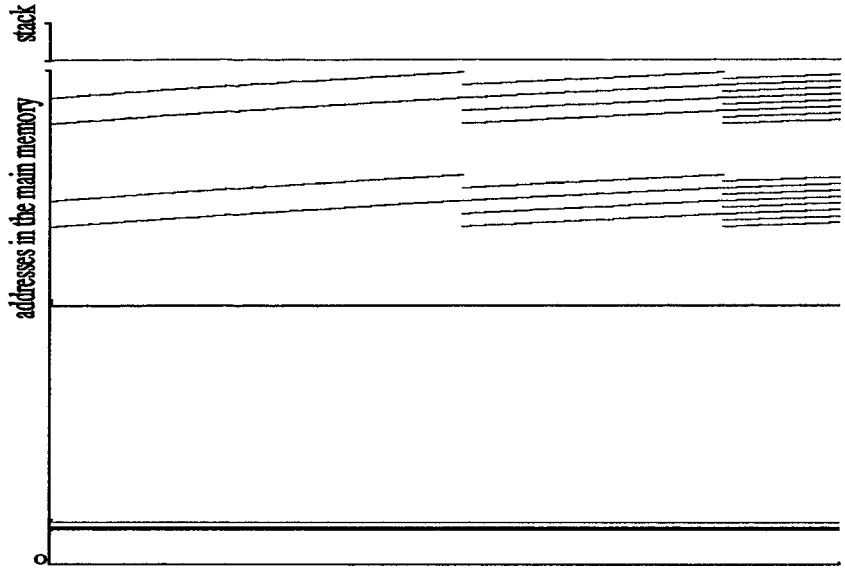
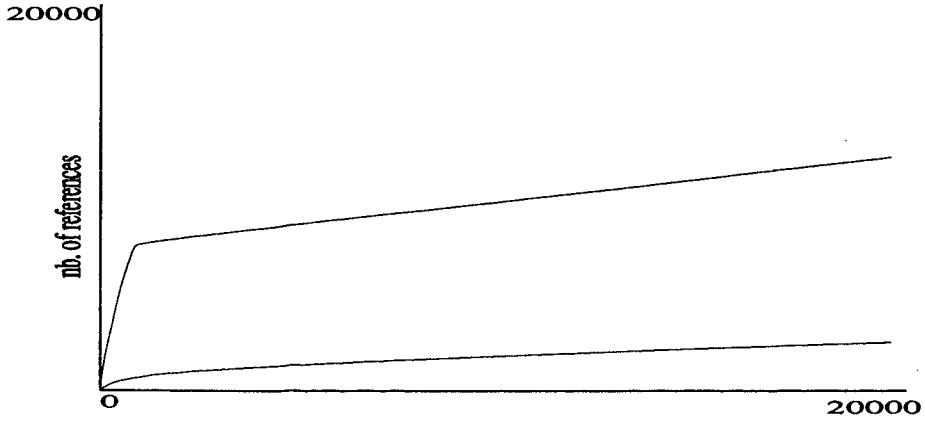
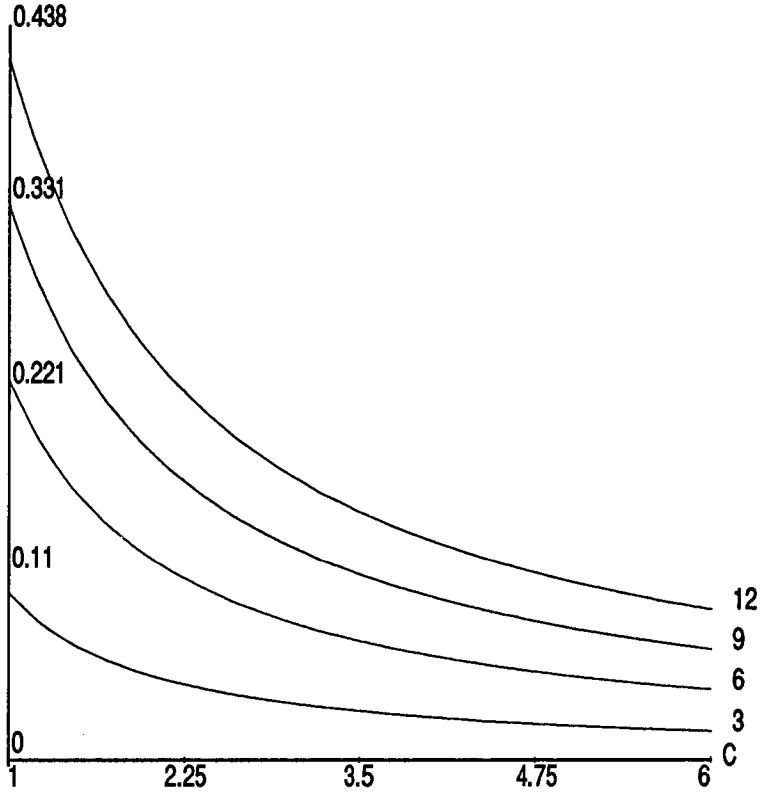


figure-3 10^6 references of a fortran fast fourier transform



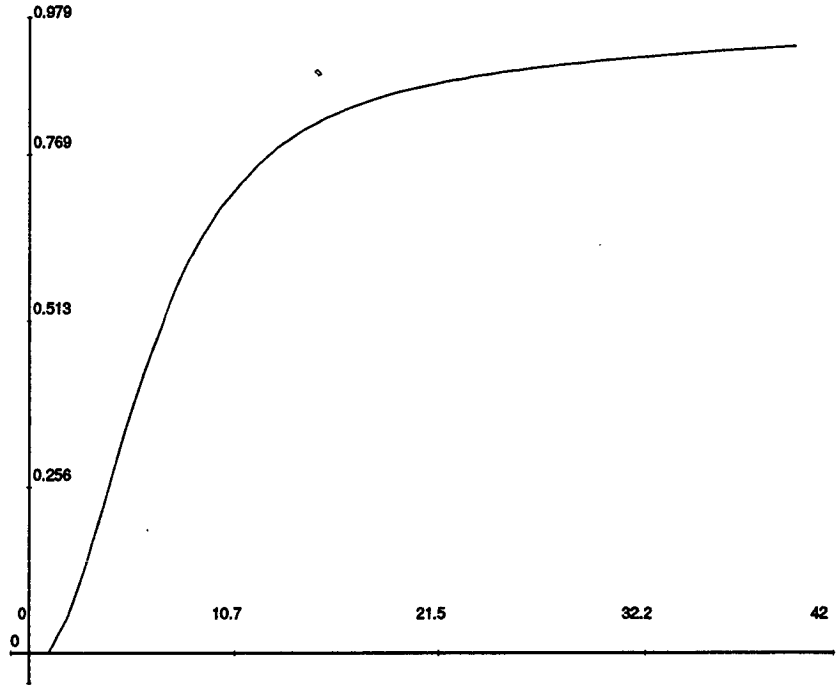
Sizes of working sets as a function of time for the FFT, the upper curve is the real one, the lower curve is the working set with an IRM model for these references.

Influence of the size of the cache



Miss ratio with the size of the cache as parameter for $3, 6, 9, 12$ elementary accesses whose length has the density $h(x) = \frac{60}{7} \times (4/5\, 1_{[0,1/8]}(x) + 1/5\, 1_{[1/4,1/3]}(x))$ (a proportion of 4 "small" intervals for a large one). Notice the sharp decrease as the size of the cache increases from 1 to 2 for 6 and 12 accesses. For these examples it is rewarding to increase the size of the cache to 3.5, after the gain is less significant.

Influence of the fragmentation of the accessed regions of main memory



Miss ratio with respect to the number of accessed regions. Let L be some random variable with density $h(x) = 15 \times (4/5\ 1_{[0,1/16]}(x) + 1/5\ 1_{[1/4,1/3]}(x))$. For n intervals the miss ratio is calculated for sizes with the distribution of L/n and $C = 1$, so that the mean occupied size of the main memory is constant with n.

More generally let D_n be the average miss ratio for δn intervals whose size has the same law as L/n, where L is some random variable with distribution $F(dx)$. Then the mean occupied space in the main memory is $\delta \int x\, F(dx)$ for any n, using the first proposition of section 4, it is easy to see that

$$D_n \to 1 - \int_0^C e^{-\frac{\delta}{C} \int_0^C x \wedge y\, F(dy)}\, F(dx)$$

as $n \to +\infty$

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique